[포팅메뉴얼]

💡 배포 순서
1. Map Server, Article Server, User Server 도커파일 생성
2. 젠킨스 도커 이미지 pull 받아서 컨테이너로 실행시킴
3. 젠킨스 관리자 계정 생성, 필요한 플러그인 설치 (Docker, Docker Pipline, GitLab, Generic Webhook Trigger, Publish over SSH
4. 젠킨스 설정에서 gitlab - credentials - GitLab API token add 하고 깃랩에서 생성한 토큰 넣어줌
5. 프로젝트 설정에서 빌드 트리거 - Build when a change is push to GitLab 체크
6. 프로젝트 설정에서 빌드 트리거 - 고급에서 시크릿 토큰 생성
7. 깃랩 webhook 설정으로 가서 젠킨스 프로젝트 설정에 있는 웹훅 url이랑 시크릿 토큰 넣고 트리거 될 브랜치명 작성함
user 브랜치, article 브랜치, location 브랜치 각각 배포
8. 각 서비스별 Execute shell 작성
9. Map Server nginx con.f 및 default에서 프록시 및 경로 설정
10. Map Server nginx Certbot 설치 (SSL 인증)

NGINX (Location-Server)

`/etc/nginx/sites-enabled/default`

```
upstream location-server {
        server localhost:8080;
}
upstream article-server {
        server 3.37.17.94:8080;
}
upstream user-server {
        server 3.34.239.211:8080;
}
upstream chat-server {
        server 3.37.17.94:8000;
}


server {

        root /var/www/html;

        server_name k8a302.p.ssafy.io;

        listen [::]:443 ssl ipv6only=on; # managed by Certbot
        listen 443 ssl; # managed by Certbot
        ssl_certificate /etc/letsencrypt/live/k8a302.p.ssafy.io/fullchain.pem; # managed by Certbot
        ssl_certificate_key /etc/letsencrypt/live/k8a302.p.ssafy.io/privkey.pem; # managed by Certbot
        include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
        ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot


        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_set_header X-Forwarded-Proto https;
        proxy_headers_hash_bucket_size 512;
        proxy_redirect off;


        location /api/users {
                client_max_body_size 100M;
                proxy_pass http://user-server;
        }
        location /api/follow {
                client_max_body_size 100M;
                proxy_pass http://user-server;
        }
        location /api/following {
                client_max_body_size 100M;
                proxy_pass http://user-server;
        }
        location /api/notifications {
                client_max_body_size 100M;
                proxy_pass http://user-server;
        }
        location /api/articles {
                client_max_body_size 100M;
                proxy_pass http://article-server;
        }
```

```
        location /api/map {
                client_max_body_size 100M;
                proxy_pass http://location-server;
        }
        location / {
                client_max_body_size 100M;
                proxy_pass http://article-server;
                proxy_http_version 1.1;
                proxy_set_header Upgrade $http_upgrade;
                proxy_set_header Connection "upgrade";
        }
        #location / {
        #        #try_files $uri $uri/ =404;
        #        proxy_pass http://user-server;
        #}


}


#server {
#       listen 80;
#       listen [::]:80;
#
#       server_name example.com;
#
#       root /var/www/example.com;
#       index index.html;
#
#       location / {
#               try_files $uri $uri/ =404;
#       }
#}
server {
    if ($host = k8a302.p.ssafy.io) {
        return 301 https://$host$request_uri;
    } # managed by Certbot


        listen 80 default_server;
        listen [::]:80 default_server;

        server_name k8a302.p.ssafy.io;
        location / {
                rewrite ^(.*)k8a302.p.ssafy.io:443$1 permanent;
        }
    return 404; # managed by Certbot


}
```

DOCKER

`Docker Setting`

```
sudo apt-get update

sudo apt-get install \
  ca-certificates \
  curl \
  gnupg

sudo install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
sudo chmod a+r /etc/apt/keyrings/docker.gpg

echo \
  "deb [arch="$(dpkg --print-architecture)" signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \
  "$(. /etc/os-release && echo "$VERSION_CODENAME")" stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

sudo apt-get update

sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin

sudo docker run hello-world
```

`Jenkins Setting`

```
docker run -d -p 9090:8080 --name jenkins -v /var/run/docker.sock:/var/run/docker.sock -v /home/jenkins:/var/jenkins_home -p 50000:500

docker exec -itu 0 jenkins /bin/bash

rm /etc/apt/sources.list.d/docker.list
apt-get remove docker docker-engine docker.io containerd runc

apt-get update

apt-get install \
    ca-certificates \
    curl \s
    gnupg \
    lsb-release

mkdir -p /etc/apt/keyrings

curl -fsSL https://download.docker.com/linux/debian/gpg | gpg --dearmor -o /etc/apt/keyrings/docker.gpg

echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/debian \
  $(lsb_release -cs) stable" | tee /etc/apt/sources.list.d/docker.list > /dev/null

apt-get update

apt-get install docker-ce docker-ce-cli containerd.io docker-compose-plugin

chmod 666 /var/run/docker.sock
```

MYSQL

`Setting`

```
sudo apt update

sudo apt upgrade

sudo apt install mysql-server

mysql -u root -p

sudo mysql

ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password by 'USER_PWD’;

mysqld.cnf bind-address Setting

service mysql restart
```

JENKINS

1. Location-Server

`/home/jenkins/workspace/areastory/Back-end/location/Dockerfile`

```
FROM openjdk:11-jdk
ARG JAR_FILE=build/libs/*.jar
#COPY ${JAR_FILE} ./app.jar
COPY ${JAR_FILE} app.jar
#EXPOSE 8099
ENTRYPOINT ["java", "-jar", "app.jar"]
```

`execute shell`

```
cd Back-end/location
chmod +x gradlew
./gradlew build

docker stop location-server
docker rm location-server
docker rmi location-server

docker build -t location-server .
docker run -d -p 8080:${Spring_Server_Port} --name location-server \
-e DB_URL=jdbc:mysql://${DB_URL}:${DB_Port}/${DB_Schema}\
-e DB_USER=${DB_USER}\
-e DB_PASSWORD=${DB_PASSWORD} \
location-server
```

2. Article-Server

`/home/jenkins/workspace/areastory/Back-end/Article/Dockerfile`

```
FROM openjdk:11-jdk
ARG JAR_FILE=build/libs/*.jar
#COPY ${JAR_FILE} ./app.jar
COPY ${JAR_FILE} app.jar
#EXPOSE 8099
ENTRYPOINT ["java", "-jar", "app.jar"]
```

`execute shell`

```
cd Back-end/Article
chmod +x gradlew
./gradlew build

docker stop article-server
docker rm article-server
docker rmi article-server

docker build -t article-server .
docker run -d -p 8080:${Spring_Server_Port} --name article-server \
-e DB_URL=jdbc:mysql://${DB_URL}:${DB_Port}/${DB_Schema} \
-e DB_USER=${DB_USER} \
-e DB_PASSWORD=${DB_PASSWORD} \
-e BUCKET_NAME=${BUCKET_NAME}\
-e AWS_ACCESS_KEY=${AWS_ACCESS_KEY} \
-e AWS_SECRET_KEY=${AWS_SECRET_KEY} \
article-server
```

3. User-Server

`/home/jenkins/workspace/areastory/Back-end/User/Dockerfile`

```
FROM openjdk:11-jdk
ARG JAR_FILE=build/libs/*.jar
#COPY ${JAR_FILE} ./app.jar
COPY ${JAR_FILE} app.jar
#EXPOSE 8099
ENTRYPOINT ["java", "-jar", "app.jar"]
```

`execute shell`

```
cd Back-end/User
chmod +x gradlew
./gradlew build

docker stop user-server
docker rm user-server
docker rmi user-server

docker build -t user-server .
```

```
docker run -d -p 8080:${Spring_Server_Port} --name user-server \
-e DB_URL=jdbc:mysql://${DB_URL}:${DB_Port}/${DB_Schema} \
-e DB_USER=${DB_USER} \
-e DB_PASSWORD=${DB_PASSWORD} \
-e BUCKET=${BUCKET_NAME} \
-e ACCESS-KEY=${AWS_ACCESS_KEY} \
-e SECRET-KEY=${AWS_SECRET_KEY} \
user-server
```

Spring Boot

1. Location-Server

- **Application.yml**

```
server:
  port: 5001
  servlet:
    context-path: /
    encoding:
      charset: UTF-8
      enabled: true
      force: true

spring:
  profiles:
    include: API-KEY
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    username: ${DB_USER}
    url: ${DB_URL}
    password: ${DB_PASSWORD}
  jpa:
    database: mysql
    hibernate:
      ddl-auto: update
    properties:
      hibernate:
        jdbc:
          time_zone: Asia/Seoul
    show-sql: true
    generate-ddl: true
  servlet:
    multipart:
      max-file-size: 100MB
      max-request-size: 100MB

cloud:
  aws:
    s3:
      bucket: ${BUCKET}
    credentials:
      access-key: ${ACCESS-KEY}
      secret-key: ${SECRET-KEY}
    region:
      static: ap-northeast-2
    stack:
      auto: false
```

2. Article-Server

- **Application.yml**

```
server:
  servlet:
    contextPath: /
    encoding:
      enabled: true
      force: true
      charset: UTF-8
  port: 5002
spring:
```

```
  jpa:
    hibernate:
      ddl-auto: update
      use-new-id-generator-mappings: false
    properties:
      hibernate:
        show_sql: true   # System out
        format_sql: true
        jdbc:
          time_zone: Asia/Seoul
    open-in-view: false
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    hikari:
      password: ${DB_PASSWORD}
      username: ${DB_USER}
    url: ${DB_URL}
  servlet:
    multipart:
      max-file-size: 100MB
      max-request-size: 200MB
  data:
    web:
      pageable:
        one-indexed-parameters: true

logging:
  level:
    com:
      amazonaws:
        util:
          EC2MetadataUtils: error

#aws
cloud:
  aws:
    credentials:
      secret-key: ${AWS_SECRET_KEY}
      access-key: ${AWS_ACCESS_KEY}
    s3:
      bucket: ${BUCKET_NAME}
    region:
      static: ap-northeast-2
      auto: false
    stack:
      auto: false
```

3. User-Server

- **Application.yml**

```
server:
  servlet:
    contextPath: /
    encoding:
      enabled: true
      force: true
      charset: UTF-8
  port: 5003
spring:
  jpa:
    hibernate:
      ddl-auto: update
    open-in-view: false
    properties:
      hibernate:
        show_sql: true
        format_sql: true
        jdbc:
          time_zone: Asia/Seoul
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    hikari:
      password: ${DB_PASSWORD}
      username: ${DB_USER}
    url: ${DB_URL}

  mvc:
    pathmatch:
      matching-strategy: ANT_PATH_MATCHER
```

시나리오

1. 로그아웃상태
2. 앱 화면 → 로그인
3. 맵에서 확대 / 축소 후 현재 위치 찍기
4. 아무 지역 선택 후 해당지역 SNS 로 바로 가기
5. 대충 스크롤로 보여주고 SNS(탭) 페이지로 가기
6. 검색 기능 (서울특별시 서초구) 쓰기
7. 현재 위치버튼으로 바로 현재위치 바뀌는거 보여주기
8. 실시간 채팅으로 들어가 시연 ⇒ 수민폰으로 대기하다 같이 채팅 좀 해주기
9. 정렬 최신순 바꾸고 스크롤 좀 내리기
10. 이때, 좋아요 0개인 게시글 찾아서 좋아요 누르고, 좋아요 목록(좋아요개수)으로 들어가기
11. 다시 나와서, 댓글 누르고 생성 → 수정 → 삭제까지 보여주고 나오기
12. 내 게시글 찾아서 수정하고, 삭제하고 게시글생성(탭)으로 이동
13. 생성하고 내용치고 공개/비공개 왔다갔다하다가 공개로 생성(자동으로 follow로 이동함)
14. 스크롤 좀 내려서 보여주다 위의 프로필(팔로우 아닌유저) 누르면 마이페이지 가는거 보여줌
15. 팔로우 신청하기
16. 우측 상단옵션버튼 → 신고하기 탭해서 나가기
17. 마이페이지 이동 → 프로필 이미지 선택 → 이미지 변경, 닉네임수정
18. 팔로우 숫자 클릭 → 팔로잉 선택 → 팔로잉 취소 → 팔로워 선택 → 팔로잉취소, 휴지통선택 → 확인 → 검색탭 → 최현제 검색 → 팔로워 팔로잉 하고싶은거하기
19. 뒤로가기, 뒤로가기
20. 게시글 선택 → 게시글 상세 보여주고 뒤로가기
21. 마이앱 보여주고 이동과 축소확대, 지도사진 탭
22. 상단 알림버튼 → 설명 → 아무거나 읽음처리, 삭제하고, 모든 알림 체크, 휴지통 삭제, 뒤로가기
    a. 종류(내게시글 좋아요, 내게시글 댓글 추가, 내 댓글 좋아요 → 해당 게시글로 이동, 팔로우는 해당 유저의 마이페이지로 이동)
23. 상단 옵션눌러서 서비스탈퇴눌러보고 취소하고 로그아웃